Electronics & Controls

Patrick Fairbank & Jay Shah

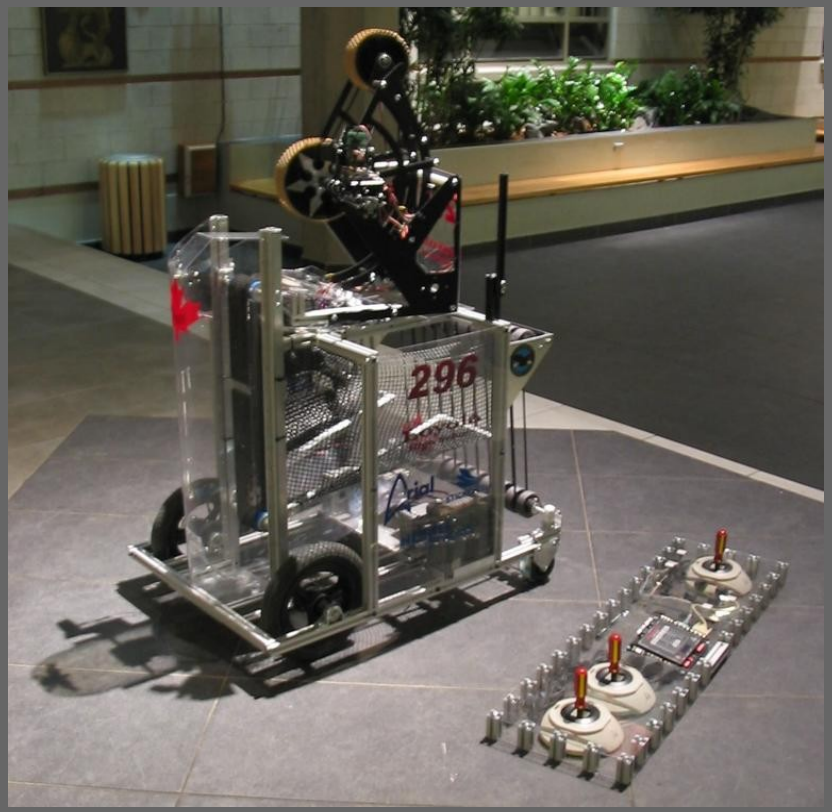
University of Toronto – November 17, 2007

Overview

- Control system
- Wiring
- Introduction to programming
- Advanced programming
- Sensors
- Autonomous programming
- Custom controls

About Patrick

- 7 years in FIRST
- 2001 – 2006: Team 296 – Northern Knights, Montreal, QC
- 2007 – present: Team 1503 – Spartonics, Niagara Falls, ON

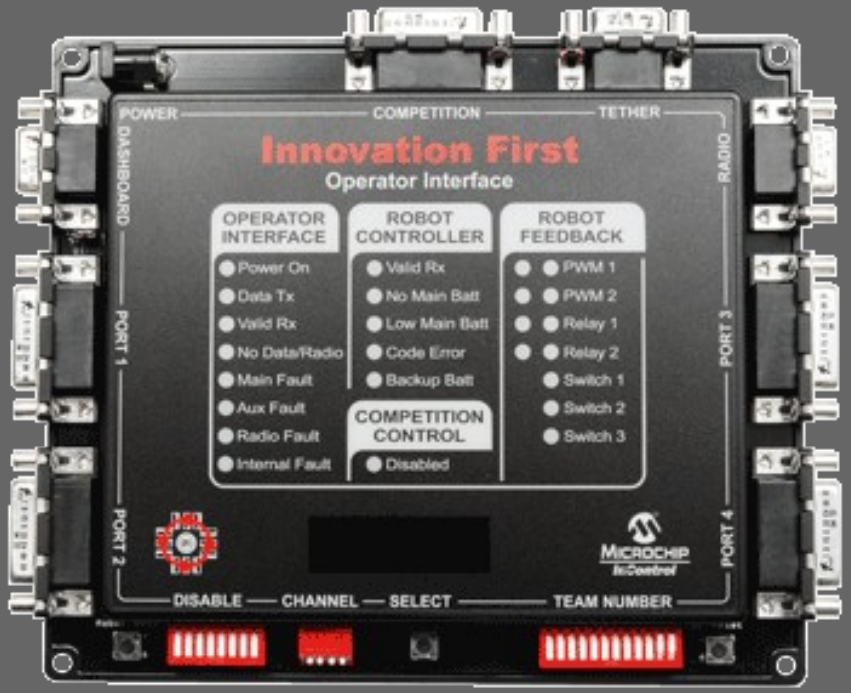


About Jay

- 6th year in FIRST
- Home Team: Crescent Robotics (Team 610)
- 2nd Year University of Waterloo, Mechatronics Engineering

Control System

Operator Interface (OI)



- 4 joystick ports
- Competition, tether and radio ports
- 4-digit LCD display
- Team number DIP switches

Control System

Robot Controller (RC)

- Contains a master microprocessor and a user microprocessor
- User processor is a Microchip PIC 18F8722, programmable in C
- 16 PWM and 8 relay outputs
- 16 analog inputs
- 16 digital inputs/outputs, including 6 interrupt inputs

Control System

Radio Modems

- 40 channels
- Operate on the 900 MHz spectrum

Control System

Speed Controller (Victor 884)

- Used to control motors with variable speed, forward and reverse
- Use pulse-width modulation (PWM)
- Maximum current of 40 Amps

Control System

Relay (Spike)

- 3 settings: forward, reverse, or off
- Used to control smaller motors
- Used to control the air compressor and pneumatic valves
- Maximum current of 20 Amps

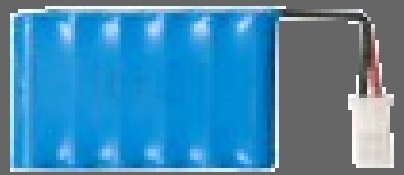
Wiring

Main battery

- 12 V, 18 AH sealed lead-acid (SLA) battery
- Used to power motors, electronics, pneumatics

Backup battery

- 7.2 V NiCd battery
- Used to power servos and some sensors
- Powers the robot controller when the main battery fails



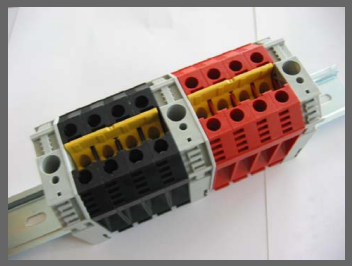
Wiring

Main circuit breaker

- 120 Amp limit

Distribution block

- Splits positive and negative leads from the battery between the various breaker panels



Circuit breaker panels

- Positive leads to all devices must pass through a circuit breaker
- 20, 30 and 40 Amp auto-resetting circuit breakers

Wiring

Circuit breaker rules

Application	Circuit breaker
Victor speed controller	20A, 30A or 40A
Spike relay	20A
Robot controller	20A
Custom circuits	20A

Wire gauge rules

Application	Wire gauge
40A circuit breaker	12 AWG or larger
30A circuit breaker	14 AWG or larger
20A circuit breaker	18 AWG or larger
Sensors & custom circuits	24 AWG or larger

Types of Wire

Voltage Drop

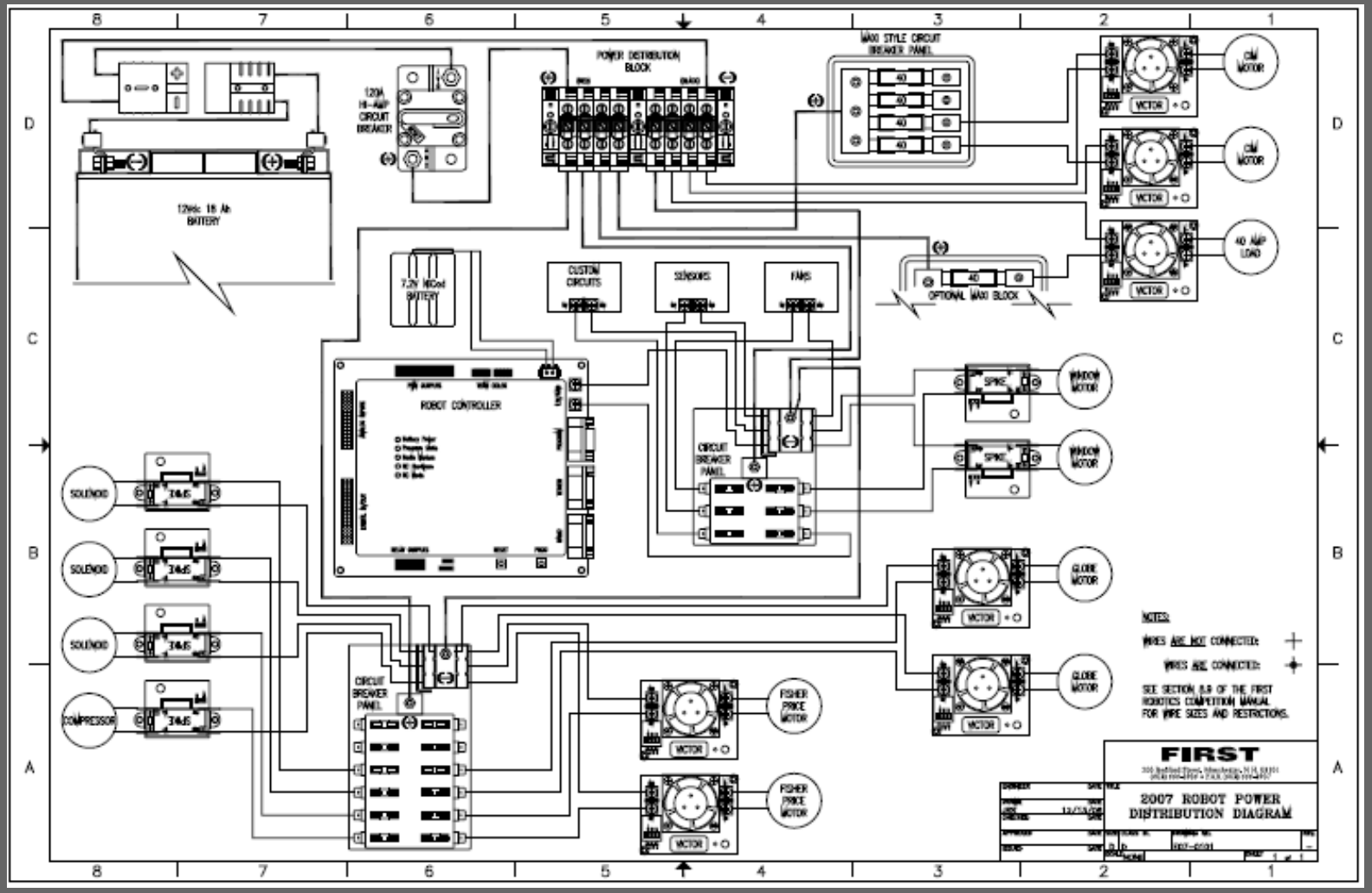
- Some power is always lost in the medium used to transport it. The metric used to quantify this is the voltage drop produced by the increased resistance that the wire offers in addition to the main load.

- Scenario: Motor on upper arm segment of robot – easily 8 ft of wire from Victor to motor.
  - What do you think the difference in the voltage that the motor sees is, if you use 10 AWG, versus 14 AWG.
    - @ 14 AWG, gives a 0.618 V drop -> equivalent to 18.54 watts (5%)
    - @ 10 AWG, gives a 0.243 V drop -> equivalent to 7.29 (2%) watts lost
    - What about the rest of the stuff in-between the victor and the battery!?

- Side point: keep in mind while designing that you are never dealing with the 'ideal' scenario

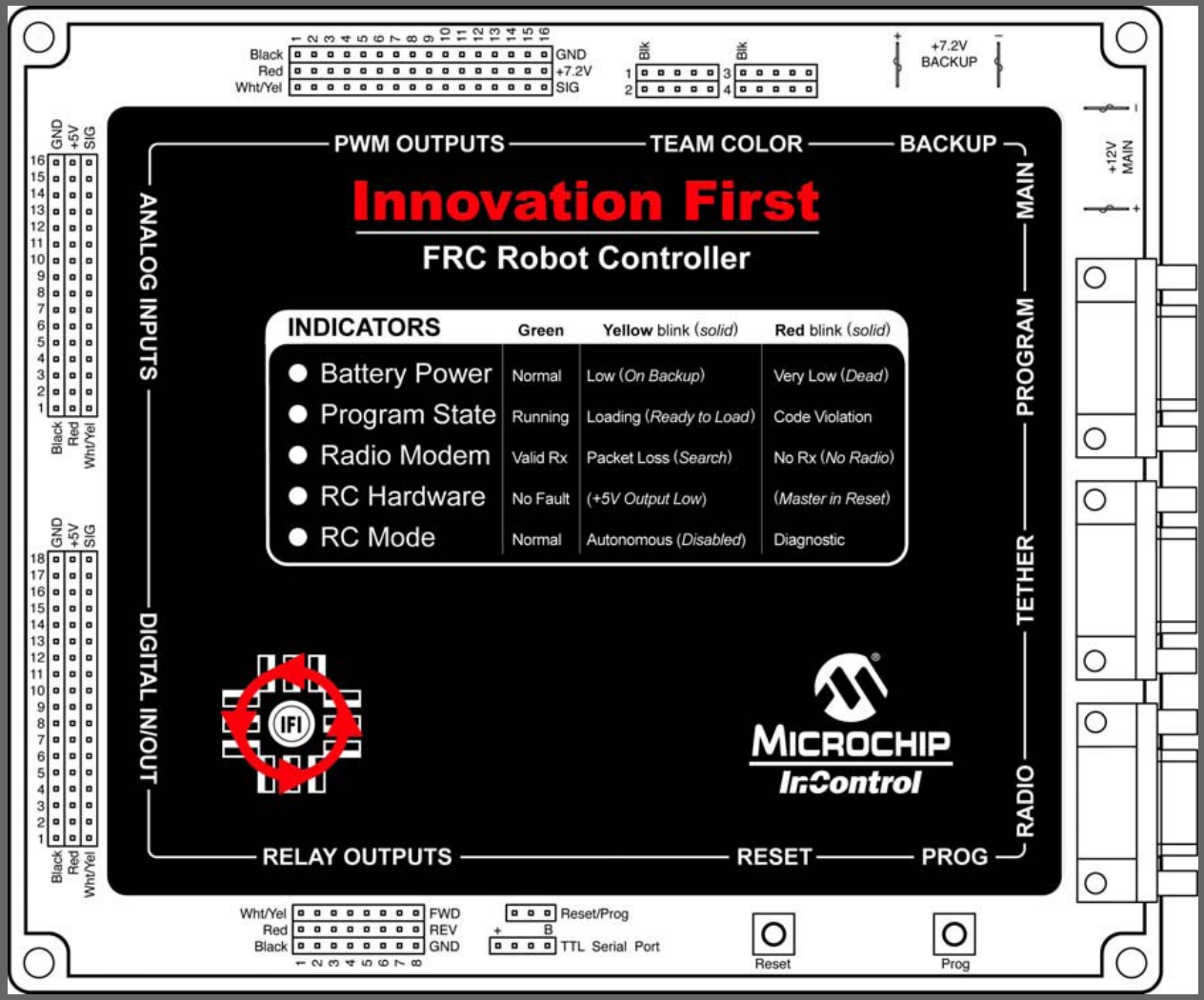
Wiring

Power distribution



Control System

Input & output signal wiring



Programming Basics

MPLAB IDE & MCC18 compiler

- Provided by Microchip
- Tools for editing, compiling and linking code

IFI Loader

- Provided by Innovation FIRST
- Tool for downloading code to the RC and viewing output from the RC

Programming Basics

Default code

- Provided by Innovation First
- Contains routines for interfacing with the master microprocessor
- Provides aliases for input and output variables (ifi_aliases.h)
- Provides functions for debugging and performing common operations
- Provides five principal interfaces for custom code:
  - The initialization sequence (user_routines.c)
  - The "slow loop" (user_routines.c)
  - The "fast loop" (user_routines_fast.c)
  - The autonomous loop (user_routines_fast.c)
  - The interrupt handler (user_routines_fast.c)

Programming Basics

Input variable types

Type	# of bits	Range	Variables
OI analog	8	0 - 254	p#_x p#_y p#_wheel p#_aux
OI digital	1	0: open circuit 1: closed circuit	p#_sw_trig p#_sw_top p#_sw_aux1 p#_sw_aux2
RC analog *	10	0 – 1023	rc_ana_in##
RC digital	1	0: closed circuit 1: open circuit	rc_dig_in##

* RC analog values are retrieved using the Get_Analog_Value(rc_ana_in##) function

Programming Basics

Output variable types

Type	# of bits	Range	Variables
RC PWM	8	0 - 254	pwm##
RC relay forward	1	1: on 0: off	relay#_fwd
RC relay reverse	1	1: on 0: off	relay#_rev
RC digital	1	1: on 0: off	rc_dig_out##

Programming Basics

The printf function

- Allows the user to send feedback from the RC to a computer, through the programming cable
- Feedback is displayed in the IFI Loader Terminal Window
- Use is similar to that of the ANSI-C printf function

Syntax:

printf(const char *format string[, var1][, var2]…*);

Example:

```
unsigned char foo = 3;
unsigned char bar = 5;
printf("Foo = %d, Bar = %4d\r", (int)foo, (int)bar);    // Prints "Foo = 3, Bar =
    5"
```

Programming Basics

Some basic programming operations

- Creating aliases for inputs and outputs
- Mapping inputs to outputs for driver control
- One-joystick drive
- Creating a dead band for joysticks
- Outputting to the OI LEDs and LCD

Advanced Programming

Interrupts

- Triggered when certain hardware-related events occur:
  - Timer reaches certain number
  - The serial port sends or receives data
  - One of the interrupt inputs on the RC has changed state
- Current execution is stopped and the interrupt handler is run
- Faster and more reliable than polling inputs to see if they have changed

To use:

- The interrupt's priority must be set to low
- The interrupt's flag must be cleared initially and whenever it is triggered
- The edge on which to interrupt must be set
- The interrupt must be enabled

Advanced Programming

EEPROM

- Electrically-Erasable Programmable Read-Only Memory
- Non-volatile (keeps value when power is off)
- Good for storing settings like joystick or sensor calibrations
- The RC has 1024 Kb of EEPROM

Reading:

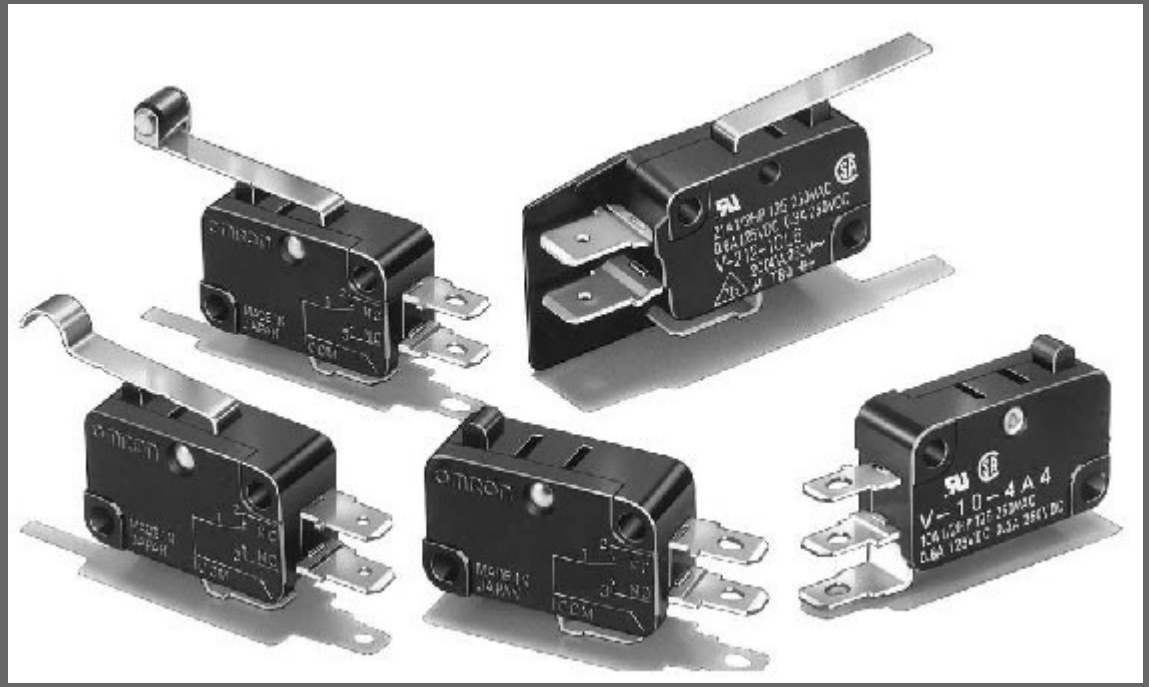
- Store the EEPROM address in EEADRH:EEADR
- Set up the EEPROM control registers
- Set EECON1bits.RD to 1 to execute the read
- The data is now stored in EEDATA

Writing:

- Store the EEPROM address in EEADRH:EEADR
- Store the data to be written in EEDATA
- Set up the EEPROM control registers
- Set EECON1bits.WR to 1 to execute the write
- Wait for the write to finish

Sensors

Limit switch



- Digital sensor
- Used mostly to limit the motion of components within a robot
- 3 terminals – Ground, Normally Closed (NC) and Normally Open (NO)

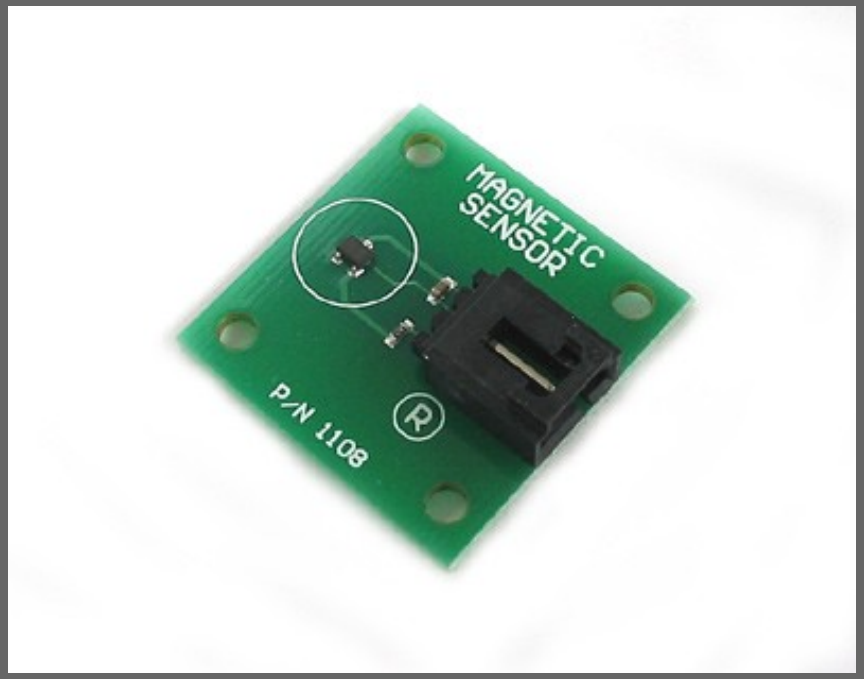
Sensors

Potentiometer

- Analog sensor
- Variable resistor
- Used to measure angular position
- 3 terminals – ground, input voltage and signal

Sensors

Hall effect sensor (gear tooth counter)



- Digital sensor – used with interrupts
- Used to measure distance/speed of rotation
- 3 terminals – power in, ground, signal

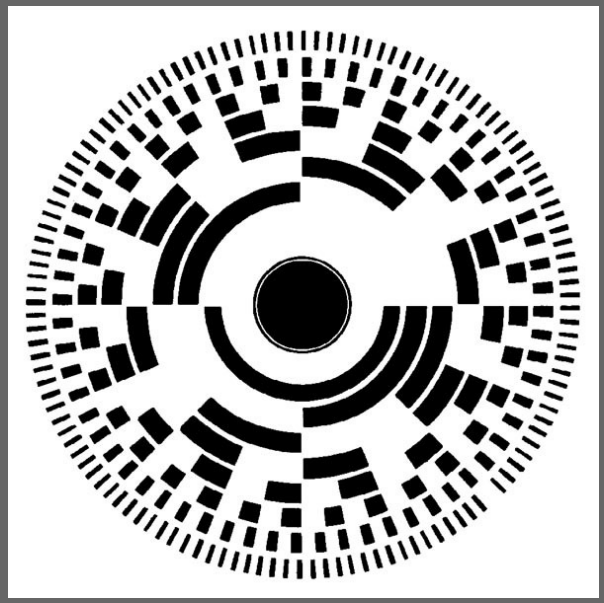
Sensors

Quadrature encoder

- Digital sensor – used with interrupts
- Used to measure direction and distance/speed of rotation
- 4 terminals – power in, ground, A channel, B channel

Sensors

Absolute encoder



- Analog sensor
- A “continuous-rotation” potentiometer
- Used to measure angular position/speed
- 3 terminals – power in, ground, signal

Sensors

Gyroscope (yaw rate sensor)

- Analog sensor
- Measures rotational velocity about the vertical axis
- Used to determine robot direction
- 3 terminals – power in, ground, signal

Autonomous Programming

What is autonomous mode?

- Controlled through the Operator Interface's competition port
- All operator inputs are disabled
- The User_Autonomous_Code function in user_routines_fast.c is run

Types of autonomous programs

- Time-based
- Relative positioning
- Absolute positioning
- Combinations of the above

Autonomous Programming

Time-based autonomous programs

- Based on the slow loop's 26 ms cycle time
- A counter variable is incremented each cycle to track time
- Based on the counter's current value, different actions are performed
- Not always accurate, since it assumes motor speeds are always constant

Autonomous Programming

Relative positioning autonomous programs

- Based on keeping track of how far the robot or its sub-mechanisms have traveled from their starting points
- Makes use of sensors – encoders, gyroscopes, Hall effect sensors, etc.
- Can be more accurate than time-based autonomous modes, but not perfectly accurate as distance measurements tend to drift over time
- Often makes use of PID control to meet position and/or speed targets with precision

PID control

- An algorithm to determine what value to output to an actuator:
  - **P**roportional: The error, or how far the measurement of something is to its goal
  - **I**ntegral: The accumulation of the error over time
  - **D**erivative: How fast the error is increasing or decreasing

Autonomous Programming

Absolute positioning autonomous programs

- Based on using things external to the robot to guide its trajectory
  - Green lights (CMUcam)
  - White lines on the carpet (light sensor)
  - Solid playing field objects (Ultrasonic or IR rangefinders)
  - GPS
- More accurate than time-based or dead reckoning autonomous programs
- More difficult to implement
- Also often uses PID

Combination autonomous programs

- Most commonly used, because they provide a balance between accuracy and ease of implementation

Custom Controls

Interfacing with the joystick ports

		C Code Information		Disable
Pin	Function	Variable type	Variable Name	Dipswitch
3	X-Axis	Analog (byte)	p1_x	
6	Y-Axis	Analog (byte)	p1_y	
13	Wheel	Analog (byte)	p1_wheel	
11	Aux Analog	Analog (byte)	p1_aux	
2	Joystick Trigger Switch	Digital (bit)	p1_sw_trig	SW01 [1]
7	Joystick Thumb Switch	Digital (bit)	p1_sw_top	SW02 [1]
10	Aux Switch1	Digital (bit)	p1_sw_aux1	SW03 [1]
14	Aux Switch2	Digital (bit)	p1_sw_aux2	SW04 [1]
15	Robot Feedback LED driver [2]	Output (bit)	Pwm1_green	
8	Robot Feedback LED driver [2]	Output (bit)	Pwm1_red	
9	Robot Feedback LED driver [2]	Output (bit)	Pwm2_green	
5	Robot Feedback LED driver [2]	Output (bit)	Pwm2_red	
1	+5V Aux (Fuse F2) [3,4,5]			
4	Ground			
12	Ground			

Custom Controls

Custom Controls

USB Chicklet

- Accepts input from a USB device and converts to a 15-pin joystick port output
- Supported devices include: Logitech joysticks, Xbox controller, NASCAR steering wheel, IntelliMouse

Useful Links

- www.ifirobotics.com
  - User guides for RC, OI, Victor and Spike
  - Default code
- www.kevin.org/frc
  - Code libraries for using encoders, gyros and the EEPROM
- www.usfirst.org
  - Power distribution diagram
  - Electronics & wiring rules
- www.microchip.com
  - PIC 18F8722 datasheet
- www.chiefdelphi.com
  - A web forum for general discussion of FIRST robotics; many electrical & programming-related whitepapers and a good place to go for help if you get stuck